MEMORIAL UNIVERSITY OF NEWFOUNDLAND

DEPARTMENT OF MATHEMATICS AND STATISTICS

A2 **MATH 6201** Due: Feb. 13, 2017

Hand in your solutions at the beginning of class on Monday Feb. 13, 2017 (send all codes used to `abihlo@mun.ca`).

1. Explicitly verify the values of the Runge–Kutta matrix (a_{ij}) and of the weights b_i, $i, j \in \{1, 2\}$ for the two-stage Gauss–Legendre Runge–Kutta method given in class.

2. Implicit Runge–Kutta methods (as well as all other implicit numerical schemes) require the solution of a (possibly nonlinear) system of algebraic equations at each time step. There are several methods that can be used for this purpose, with two important ones being fixed-point iteration and Newton's method. In fixed point iteration one recasts the algebraic system $\mathbf{f}(\mathbf{x}) = 0$ in the equivalent form $\mathbf{x} = \mathbf{g}(\mathbf{x})$ and applies the iteration

$$\mathbf{x}^{n+1} = \mathbf{g}(\mathbf{x}^n), \quad n = 0, 1, \ldots .$$

This procedure converges provided that

$$\left|\left|\frac{\partial \mathbf{g}}{\partial \mathbf{x}}\right|\right| < 1,$$

where $J_{\mathbf{g}} = \partial \mathbf{g}/\partial \mathbf{x}$ is the Jacobian of $\mathbf{g}$. Newton's method is a particular fixed point iteration for which $\mathbf{g}(\mathbf{x}) = \mathbf{x} - J_{\mathbf{f}}(\mathbf{x})^{-1}\mathbf{f}(\mathbf{x})$.

(a) Implement Newton's method in Matlab. The prototype of the function should be $[x, n] = \texttt{newton}(f, df, x0, eps)$, where f is the system $\mathbf{f}(\mathbf{x}) = 0$ that should be solved (implemented in Matlab using e.g. *anonymous functions*), *df* is the Jacobian $J_{\mathbf{f}}(\mathbf{x})$ (implemented in Matlab using e.g. *anonymous functions*), $x0$ is the initial guess for the solution $\mathbf{x}$ and *eps* is the tolerance, i.e. the iteration terminates provided that $||\mathbf{x}^{n+1} - \mathbf{x}^n|| < eps$. The output variable x then contains the vector of approximate solutions $\mathbf{x}$ of the system $\mathbf{f}(\mathbf{x}) = 0$ and n is the total number of iterations.

(b) Implement the 2-step Gauss–Legendre Runge–Kutta method using Newton's method to solve the implicit system for the two stages. The prototype of the function should be $u = \texttt{gaussLegendreRK2}(f, df, u_0, t, \Delta t)$, where f and *df* are as above, u_0 is the vector of initial conditions, t is the final integration time and Δt is the time step. We will use this method in Problem 4 to solve the famous Lorenz–1963 system.

3. In this problem and the following problem we explore the idea of embedded Runge–Kutta methods numerically. A classical embedded Runge–Kutta method is due to Fehlberg, who derived a fourth and a fifth order method (recall that for a fifth order explicit Runge–Kutta method at least six stages are required). A modification of Fehlberg's idea is due to Cash and Karp, giving the *Cash–Karp method.* This method has the embedded Butcher tableau given below.

0						
$\frac{1}{5}$	$\frac{1}{5}$					
$\frac{3}{10}$	$\frac{3}{40}$	$\frac{9}{40}$				
$\frac{3}{5}$	$\frac{3}{10}$	$-\frac{9}{10}$	$\frac{6}{5}$			
1	$-\frac{11}{54}$	$\frac{5}{2}$	$-\frac{70}{27}$	$\frac{35}{27}$		
$\frac{7}{8}$	$\frac{1631}{55296}$	$\frac{175}{512}$	$\frac{575}{13824}$	$\frac{44275}{110592}$	$\frac{253}{4096}$	
	$\frac{37}{378}$	0	$\frac{250}{621}$	$\frac{125}{594}$	0	$\frac{512}{1771}$
	$\frac{2825}{27648}$	0	$\frac{18575}{48384}$	$\frac{13525}{55296}$	$\frac{277}{14336}$	$\frac{1}{4}$

Table 1: Butcher tableau for the Cash–Karp method.

The first method is of order five, the second method is of order four. The error control mechanism is as follows: We know that the local one-step error of the fourth-order method scales as $(\Delta t)^5$, thus we have the relation

$$\left(\frac{\Delta t_0}{\Delta t_1}\right)^5 = \left|\frac{\mathcal{L}_0}{\mathcal{L}_1}\right|, \tag{1}$$

where Δt_1 and $\mathcal{L}_1$ are the time step and local one-step error at the current time step and $\mathcal{L}_0$ is the desired one-step error. If $\mathcal{L}_1$ is larger than $\mathcal{L}_0$, the above equation tells how much to decrease the step size if the current time step Δt_1 has to be rejected and the current step is to be recomputed using the new time step Δt_0. Similarly, if Δt_1 is smaller than Δt_0 this equation tells how much we can increase the time step in the next step.

The problem is that when we talk about accuracy we usually want to bound the global error rather than the local one-step error. Hence, specifying $\mathcal{L}_0$ can be tricky. Recall that for a stable numerical scheme we have the relation $|\mathcal{L}| \propto \Delta t|e|$, where e is the global error. Hence, we accept a time step provided that $|\mathcal{L}_1| \leq \Delta t_1\, tol$, where tol is a user specified tolerance. If $|\mathcal{L}_1| > \Delta t_1\, tol$ we recompute the current step setting $|\mathcal{L}_0| = \alpha \Delta t_0\, tol$ in Eq. (1), giving

$$\Delta t_0 = \Delta t_1 \left|\frac{\alpha \Delta t_1\, tol}{\mathcal{L}_1}\right|^{1/4}, \tag{2}$$

where the user specified parameter $\alpha \leq 1$ is included for safety purposes.

Note that despite the error estimate is for the fourth order method it is customary to use the fifth-order approximation for the new value u^{n+1}.

(a) Implement the Cash–Karp method in Matlab. The method should accept a generic vector-valued function $\mathbf{f} = (f_1(t,\mathbf{u}), \ldots, f_r(t,\mathbf{u}))$ (the right hand side of the system of ODEs $\mathbf{u}' = \mathbf{f}(t,\mathbf{u})$) where $\mathbf{u} = (u_1, \ldots, u_r)$ as an input variable. The prototype of the function should be $[u,t] = \texttt{ck45}(f, u_0, t, \Delta t_1, tol)$, where the input variables are as follows: f is the right-hand side of the function (implemented using e.g. *anonymous functions*), u_0 is the initial condition, t is the final integration time, Δt_1 is the initial time-step and *tol* is the error tolerance. The output variables are the vector of numerical solution values u defined at the vector of time steps t. Choose $\alpha = 0.9$.

(b) Test the numerical method for a system of ODEs with known exact solution to verify it works properly.

4. The Lorenz–1963 model is one of the most famous dynamical systems and has been investigated in countless books and thousands of papers. It reads

$$u_1' = \sigma(u_2 - u_1), \quad u_2' = u_1(\rho - u_3) - u_2, \quad u_3' = u_1 u_2 - \beta u_3,$$

where σ, ρ and β are the system paramters.

(a) Integrate the Lorenz system with the Matlab ODE solver `ode23s`. Choose $\sigma = 10$, $\beta = 8/3$ and $\rho = 50$ as systems parameters and $u_1(0) = u_2(0) = u_3(0) = 1$ as the initial conditions. Integrate up to $t = 50$.

(b) Compare the result obtained (i.e. the coordinates of the final point) with the one obtained using the Cash–Karp method and the 2-stage Gauss–Legendre Runge–Kutta method. Interpret the result.